УДК 004.37

Н.Б. Шаховська, Х.Р. Шаховська
Національний університет «Львівська Політехніка», Україна
вул. Степана Бандери, 12, м. Львів, 79013

МЕТОД АНАЛІЗУ ВІДГУКІВ КЛІЄНІВ З ПРИРОДНОМОВНИХ ТЕКСТІВ

N.B. Shakhovska, Kh.R. Shakhovska
Lviv Polytechnic National University, Ukraine
12, Stepan Bandery St., Lviv, 79013

THE CUSTOMER SENTIMENT ANALYSIS METHOD FROM NATURAL LANGUAGE TEXTS

Стаття присвячена методу аналізу текстів природною мовою, що містять відгуки клієнтів. Метод відрізняється від існуючих комбінацією різних типів векторизатора та уведенням ієрархії компонентів. Послідовність застосування різних векторизаторів дає змогу будувати ієрархію ознак та маркерів. Використання методу опорних векторів та острівної кластеризації з подальшим навчання моделі для прогнозування почуттів є одним із кращих методів аналізу настроїв, як для небінарних, так і для бінарних аспектів. На основі відкритого набору даних з допомогою Python та Tablau побудовано програмний продукт для аналізу вподобань клієнтів і візуалізації результатів аналізів.

Ключові слова: сентимент-аналіз, векторизатор, природномовний текст, кластеризація

The article is devoted to the method of analysis of texts in the natural language, containing reviews of clients. The method differs from the existing combination of different types of vectorizer and the introduction of the component hierarchy. The sequencing of the use of different vectorizers allows us to build a hierarchy of features and markers. Using the reference vectors and island clustering techniques, with the subsequent training of a model for prediction of feelings, is one of the best methods for analyzing mood, both for non-binary and binary aspects. Based on open data set with Python and Tablau, a software product was developed to analyze customer preferences and visualize the results of analyzes.

Keywords: sentiment analisys, vectorizer, text in natural language, clustering

Вступ

Аналіз настроїв (сентимент-аналіз, Sentiment Analysis) допомагає розпізнати враження клієнта від компанії, товарів чи послуг. Особливо аналіз настроїв поширений у соціальних мережах, і саме з їх появою відгуки клієнтів стали аналізуватися в автоматичному режимі. Емоції клієнта, в основному, впливають на формування маркетингової стратегії [1].

Оскільки аналіз настроїв належить до методів машинного навчання без вчителя, то єдиного результату аналізу неможливо досягнути. Різноманітність результатів аналізу є засобом прийняття рішення про покращення характеристик продукту, підвищення обсягів продажів тощо.

Метою статті є використання алгоритмів NLP (обробки природної мови, Nature Language Processing) для аналізу настроїв клієнтів. Для цього використано відкритий набір даних Women E-comerce Closing Reviews (https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews) з більше ніж 23000 дописами, мову програмування Python та Tableau.

Постановка проблеми

Маємо множину документів з настроями клієнтів *D*: кожен документ *d* поданий п'ятіркою <e_i. a_{ij}, oo_{jikl}, h_k, t_l>. Ставимо перед собою завдання:

1) Знайти усі документи з *D*, у яких є згадка про сутність e_i. Сутність є продуктом, послугою, особою, подією, організацією тощо і може бути задана як (*T*, *W*), де *T* – ієрархія компонентів (або частин), субкомпонентів, а *W* – набір атрибутів e_i. Кожен компонент або підкомпонент також має свій власний набір атрибутів;
2) Знайти аспекти a_{ij} в e_i і сформувати кластери.

3) Видобути ці шматки інформації з тексту (неструктурованих даних) та сформувати базу даних із зазначенням власника h_k документа d; та часу його формування t_l;
4) Для кожного аспекти a_{ij} визначити тональність, тобто, чи належить він до позитивного, негативного чи нейтрального (oo_{jikl});
5) Сформувати усі кортежі <e_i. a_{ij}, oo_{jikl}, h_k, t_l> та класифікувати документ d.

Отже, аналіз настроїв поєднує як методи машинного навчання без вчителя (кластеризація, класифікація), так і методи машинного навчання із вчителем (класифікація).

Аналіз останніх досліджень і публікацій

Формальна постановка задачі аналізу настроїв подано у [2]. Автори пропонують використовувати SVM (Support Vector Machine) для кластеризації, але попередньо не здійснюють обробку тексту (токенізація, забирання стоп-слів), що унеможливлює використання пропонованого методу для української мови зокрема.

У [3] існуючі підходи класифікації настроїв поділено на такі класи:

- підходи, що базуються на правилах;
- підходи, що базуються на словниках (тональні словники);
- частотні методи;
- навчання без вчителя.

Підходи, базовані на правилах, дають змогу опрацьовувати тексти лише певної вузької області, а також автоматичне поповнення правил здійснюється лише на основі функціональних залежностей.

Тональні словники визначають набір правил, що визначають настрій (тональність) тексту. Потребують розміченості текстів.

Частотні методи зазвичай працюють разом з мірою TF-IDF і використовують підхід «мішка слів», що не дає змоги враховувати зворотного значення слів (наприклад, *«веселий»* у розумінні *сумний*).

Пошук настрою тексту з використанням методів навчання без вчителя потребує їх комбінації та інколи пояснення результату експертом. Проте вони є найпоширенішими, оскільки дозволяють мінімізувати недоліки попередньо поданих підходів.

У роботі [4] автором пропонується метод отримання сигнатур документа, які будуються на основі певного набору статистичних параметрів документа, обраних з міркувань стійкості до певних форм змін документа. Наприклад, приблизну кількість речень у документі можна визначити за кількістю великих літер, ком і крапок; загальна довжина тексту в символах за винятком пробілів і стоп-слів дає загальну оцінку обсягу документа тощо.

Автором методу проводилися дослідження можливості використання різноманітних параметрів текстів для опосередкованого виявлення дублювань [5]. Так, було підраховано кількість входжень у текст документа кожного символу з наступного набору: . , - _ : ; ! ? ( ) і символ пробілу. Образ документа подається у вигляді вектора розмірністю 11 елементів, і-тим компонентами якого була кількість входжень відповідного символу. В іншому тесті з документа видалялися буквено-цифрові символи, залишаючи спецсимволи, пробіли й переведення рядків.

Мета дослідження

Метою дослідження є розроблення методу аналізу відгуку клієнта шляхом комбінування різних типів векторизаторів, а також реалізація запропонованого методу. Це дасть змогу будувати цілісну маркетингову систему, яка починається від виставлення рейтингу за коментарем, яка фіналізує результати і «будує» стратегії.

Виклад основного матеріалу

Опис методу

Аналіз відгуків клієнта можна розбити на такі етапи: обробка тексту; побудова моделі; аналіз результатів.

I. Обробка тексту

Текст повинен бути попередньо оброблений для побудови моделі. Це покращить точність оцінки відгуку. Розглянемо кілька методів обробки:

Токенізація означає розбиття тексту на мінімально значущі одиниці. Це

обов'язковий етап перед будь-яким видом обробки. Базовий tokenizer (за аналогією до NLTK) розділить текст на аспекти, які використовуються в подальшому для сентимент-аналізу.

Приклади токенізованого тексту різними токенізаторами:

Токенізатор locution: *The mouse gave up. () => the/mouse/gave_up* .

Токенізатор amalgam: *If you wannabe my lover => if/you/want/to/be/my/lover/...*

Стемінг – це процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс. Результати стемінгу іноді дуже схожі на визначення кореня слова, але його алгоритми базуються на інших принципах. Тому слово після обробки алгоритмом стемінгу (стематизації) може відрізнятися від морфологічного кореня слова. Стемінг застосовується в лінгвістичній морфології та в інформаційному пошуку

Стоп-слова (Stopwords). База даних стоп-слів дає змогу видалити короткі слова, які істотно не впливають на семантику тексту, наприклад, сполучники.

Проаналізуємо кілька етапів обробки даних: для цього візьмемо один відгук і перевіримо його, як він зміниться після чистки:

```
«Yes, this is a great dress! i wasn't sure about it online because of the color combination. i think i would have preferred the gray color but it was sold out. it received very good reviews online so i thought it was worth the risk at the sale price. i am always on the hunt for great dresses at great prices (who isn't?!). once i received it and tried it on, oh wow! i love it. it is so flattering. it is a very pretty dress. i think i will wear this all the time. i am actually thinking of all the d»
```

Перший етап очищення тексту включає перетворення до нижнього регістру, видалення стоп-слів і стемінг. Код програми поданий нижче:

```
def cleaning_function(text):
    text = text.translate(string.punctuation)

    text = text.lower()
    text = text.split()

    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops and len(w) >= 3]

    text = " ".join(text)

    text = text.split()
    stemmer = SnowballStemmer('english')
    stemmed_words = [stemmer.stem(word) for word in text]
    text = " ".join(stemmed_words)
    return text
```

Отримаємо:

```
yes, great dress! sure onlin color combination. think would prefer gray color sold out. receiv good review onlin thought worth risk sale price. alway hunt great dress great price (who isn't?!). receiv tri on, wow! love it. flattering. pretti dress. think wear time. actual think
```

Наступним кроком є використання регулярних виразів. З їх допомогою та бібліотеки ми розділяємо та видаляємо найпоширеніші фрази:

```
def cleaning_function(text):
    text = text.translate(string.punctuation)

    text = text.lower()
    text = text.split()

    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops and len(w) >= 3]

    text = " ".join(text)

    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)

    return text
```

Отримаємо:

```
yes great dress sure online color
combination think would preferred
gray color sold out received good
reviews online thought worth risk
sale price always hunt great
dresses great prices who is not
received tried on wow love it
flattering pretty dress think wear
time actually thinking
```

Векторизація. Слова повинні бути закодовані як цілі числа або значення з плаваючою точкою для використання як вхідні дані для алгоритму машинного навчання, що називається вилученням ознак. Автори статті [6] описують побудову моделі векторного простору для завдання векторизації слів. Автори поєднали переваги двох підходів: локального контексту вікна і метод глобальної факторизації матриць. За результатами порівняння існуючих моделей, отримана авторами модель Glove показує непогані результати як у задачі пошуку схожих слів, так і в завданні Named Entity Recognition.

Bag-of-Words Model. Як вже було згадано вище, неможливо працювати з текстом безпосередньо при використанні алгоритмів машинного навчання. Отже, нам необхідно перетворити текст на числа. Одним із простих і ефективних способів є модель «мішок слів» або BoW. Модель проста в тому, що вона відкидає всю інформацію про порядок слів і зосереджується на входженні слів у документі. Це можна зробити, присвоївши кожному слову унікальний номер. Тоді будь-який документ, який ми бачимо, може бути закодований як вектор фіксованої довжини з довжиною словника відомих слів. Значення в кожній позиції у векторі може бути заповнене підрахунком або частотою кожного слова в закодованому документі.

Існує багато способів розширити цей простий метод, як краще уточнити, що таке слово, так і визначити, що кодувати кожне слово у векторі.

Бібліотека scikit-learn забезпечує 3 різні схеми, які ми можемо використовувати: CountVectorizer, TfidfVectorizer, HashingVectorizer.

CountVectorizer надає простий спосіб як виділити текстові документи, так і створити словник відомих слів, а також кодувати нові документи, використовуючи цей словник.

Алгоритм CountVectorizer:

1) Створити екземпляр класу CountVectorizer.
2) За допомогою функції fit () сформувати словник з одного або більше документів.
3) Викликати функцію transform () на одному або декількох документах, необхідних для кодування кожного вектора.
4) Кодований вектор повертається з довжиною всього словника і цілим числом для кількості разів кожного слова в документі.

Важливо, що той самий векторизатор може бути використаний на документах, які містять слова, не включені до словника. Ці слова ігноруються, і в отриманому векторі не наводиться кількість.

```
X.apply(cleaning_function)
c_vec = CountVectorizer()
c_vec.fit(X)
X = c_vec.transform(X)
```

```
X transformed:
  (0, 583)     1
  (0, 951)     2
  (0, 2846)    1
  (0, 10884)   1
  (0, 11133)   1
  (0, 13913)   1
```

TF-IDF vectorizer. Підрахунок слів може бути використаний для підготовки текстів до аналізу, але має ряд недоліків. Наприклад, «the» з'явиться багато разів, і їх великі значення не будуть значущими в кодованих векторах.

Альтернативою є розрахунок частот слів, і найбільш популярним методом називається TF-IDF. Це – абревіатура, що означає «частота термінів – зворотний документ», яка є складовими результуючих балів, присвоєних кожному слову. Частота термінів: підсумовує, як часто дане слово з'являється в документі. Зворотна частота документа зменшує масштаби слів, яких багато з'являється в документах.

Отже, TF-IDF – це показники частоти слів, які намагаються виділити слова, які є більш цікавими, наприклад, частими документами, але не всіма документами.

TF-IDFvectorizer шукає обернені коефіцієнти частоти документів і кодує нові документи. Крім того, якщо попередньо використати CountVectorizer, то його можна використовувати з TfidfTransformer для обчислення обернених частот документа і початку кодування документів. Процес створення, налаштування та перетворення використовується як з CountVectorizer.

```
X = clothes_cleaned['Review Text']
X.apply(cleaning_function)
tf_vec = TfidfVectorizer()
tf_vec.fit(X)
X = tf_vec.transform(X)
```

```
X transformed:
  (0, 13913)   0.4650015062801364
  (0, 11133)   0.5376963415994225
  (0, 10884)   0.492919905615551З
  (0, 2846)    0.2636953393304713
  (0, 951)     0.1991097263821536
  (0, 583)     0.3775000594849874
```

Hashing vectorizer. Підрахунок кількості слів і частота, які є основою попередньо поданих векторизаторів, можуть бути дуже корисними, але одним з обмежень цих методів є те, що словник може стати дуже великим. Це, у свою чергу, вимагатиме великих векторів для кодування документів і нав'язуватиме великі вимоги до пам'яті і уповільнюватиме алгоритми.

Покращення полягає в тому, щоб скористатися способом хешування слів для перетворення їх у цілі числа. Перевага полягає в тому, що не потрібно формувати словник, а також можна вибрати вектор довільної фіксованої довжини. Недоліком є те, що хеш є односторонньою функцією, тому не існує способу перетворити кодування назад до слова (яке може не мати значення для багатьох контрольованих завдань навчання).

HashingVectorizer реалізує цей підхід і може бути використаний також для послідовних хеш-слів з таким маркуванням та кодуванням документа.

```
X = clothes_cleaned['Review Text']
X.apply(cleaning_function)
h_vec = HashingVectorizer(ngram_range=(1, 5))
h_vec.fit(X)
X = h_vec.transform(X)
```

```
X transformed:
  (0, 88313)   0.19245008972987526
  (0, 131881)  0.19245008972987526
  (0, 154635)  0.19245008972987526
  (0, 167035)  0.19245008972987526
  (0, 180525)  -0.3849001794597505
  (0, 203648)  0.19245008972987526
  (0, 216626)  0.19245008972987526
  (0, 230285)  0.19245008972987526
  (0, 291052)  0.19245008972987526
  (0, 299433)  0.19245008972987526
  (0, 321990)  0.19245008972987526
  (0, 332294)  -
0.19245008972987526
  (0, 475715)  0.19245008972987526
  (0, 616655)  0.19245008972987526
  (0, 681602)  -
0.19245008972987526
  (0, 697710)  0.19245008972987526
  (0, 722887)  0.19245008972987526
  (0, 724616)  -
0.19245008972987526
  (0, 726496)  -
0.19245008972987526
  (0, 731228)  0.19245008972987526
  (0, 750171)  -
0.19245008972987526
  (0, 822631)  -
0.19245008972987526
  (0, 909483)  -
0.19245008972987526
  (0, 1017511) -
0.19245008972987526
```

Коли ми обробили дані, ми можемо приступати до самої моделі методом острівної кластеризації.

Основним методом острівної кластеризації є метод, який базується на використанні графа сумісної зустрічальності термів. Цей граф будується на основі визначення кореляції кожної пари термів, з яких складаються тексти, і кластеризується саме цей граф (або його частина) описаним у методі підходом. Таким чином, терми документів групуються в кластери саме на основі спільної зустрічальності. Також, завдяки цьому даний метод є стій-

ким до проблем синонімії та омонімії – терми з різним значенням з великою ймовірністю потраплять у різні кластери термів, оскільки вони будуть зустрічатись у текстах спільно з різними термами. У парі до цього для вирішення омонімії можна використовувати словники.

Вже для групування текстів у кластери на основі кластерів термів використовуються спеціальні процедури, що також описані цим методом.

Таким чином, метод острівної кластеризації текстових датасетів складається з наступних кроків:

1) Попереднє оброблення текстів з вхідної колекції документів: видалення стоп-слів, лематизація тощо.
2) Виділення з текстів множини термів, з яких вони складаються.
3) За необхідності – фільтрація отриманої множини термів (наприклад, у ситуаціях, коли відомі початкові центроїди кластерів або отримана множина є занадто великою).
4) Побудова графу кореляції термів між собою.
5) Попереднє оброблення графа і отримання його наближення.
6) Кластеризація отриманого наближення графа.
7) Розбиття документів на кластери на основі отриманих кластерів термів.

Як правило, цей метод дає кластери, що легко інтерпретувати саме на основі змісту документів, що складають ці кластери.

Розбиття документів на кластери здійснено методом опорних векторів з лінійним розділенням по гіперплощинах на основі поліноміального ядра $k(x,x') = ((x,x') + const)^d$. Вибір саме цього методу кластеризації зумовлений наявністю попередніх кроків з оброблення текстів та подання їх у векторній та графовій формах.

Послідовність кроків:

1) Перетворимо рейтинг за п'яти-бальною шкалою до бінарного вигляду, де 1-3 це 0, а 4-5 – 1.
2) Використаємо наведену вище обробку даних, а саме: cleaning function і Hashing vectorizer.
3) Розділимо дані на тренувальну і тестувальну частини як 2 до 1.
4) Натренуємо лінійну SVM.

У результаті виконання цих кроків ми отримали точність моделі (accuracy) `0.8829677609303695`.

Далі візьмемо позитивний і негативний відгук і перевіримо, як оцінить його модель:

```
Love this dress!  it's sooo
pretty.  i happened to find it in
a store, and i'm glad i did bc i
never would have ordered it online
bc it's petite.  i bought a petite
and am 5'8".  i love the length on
me- hits just a little below the
knee.  would definitely be a true
midi on someone who is truly
petite.
```

```
print(svc.predict(positive_review_transformed)[0]
)
>> 1
```

```
print(svc.predict(negative_review_transformed)[
0])
>> 0
```

Аналіз результатів

У першу чергу, проаналізуємо вміст датасету «M Women E-comerce Closing Reviews».

Структура датасету:

Clothing ID: Integer – категорична змінна, що належить до конкретної частини, що розглядається.

Age: Positive – ціла змінна віку рецензентів.

Title: змінна рядка для назви допису.

Review text: змінна рядка для тіла допису.

Rating: позитивна цілочисельна змінна для продуктової оцінки, наданої клієнтом від 1 – Найгірше до 5 – Найкраще.

Recommended IND: Двійкова змінна, де клієнт рекомендує продукт, де 1 рекомендується, 0 не рекомендується.

Positive Feedback Count: позитивне ціле число, яке визначає кількість інших клієнтів, які вважають цей огляд позитивним.

Division name: категорійна змінна, найменування продукту високого рівня поділу.

Department name: категоріальна назва, назва відділу продукту.

Class name: категорійна змінна, назви класу продукту.

Здійснимо бізнес-аналіз для Customer sentiment analysis. Розглянемо зараз кілька варіантів аналізу у Tableau. Можна аналізувати рівень якості продажів залежно від відділення, відповідно можна робити висновки щодо стратегії покращення виробництва та продажу товару.

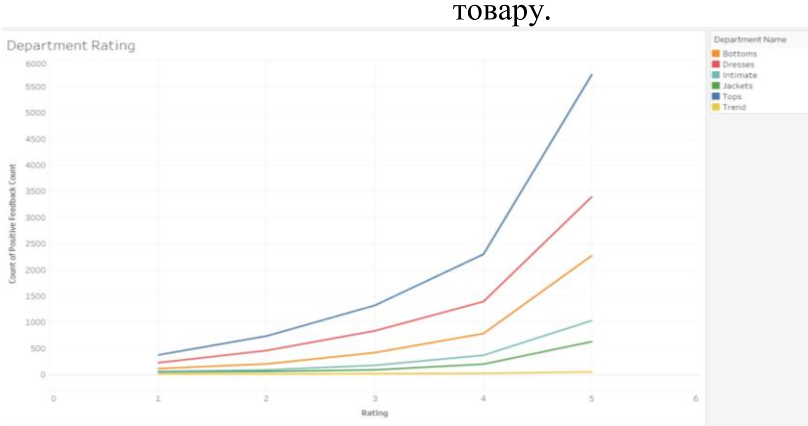


Рис. 1. Залежність рейтингу від кількості позитивних відгуків

Більше того, Tableau дозволяє робити анімовані звіти, що дозволяє дивитися інфографіку залежно від параметру. Для прикладу, аналізуватимемо вік клієнта та вплив цього фактору на кількість залишених позитивних та негативних відгуків. Результати аналізу відгуків подано на рис. 2.

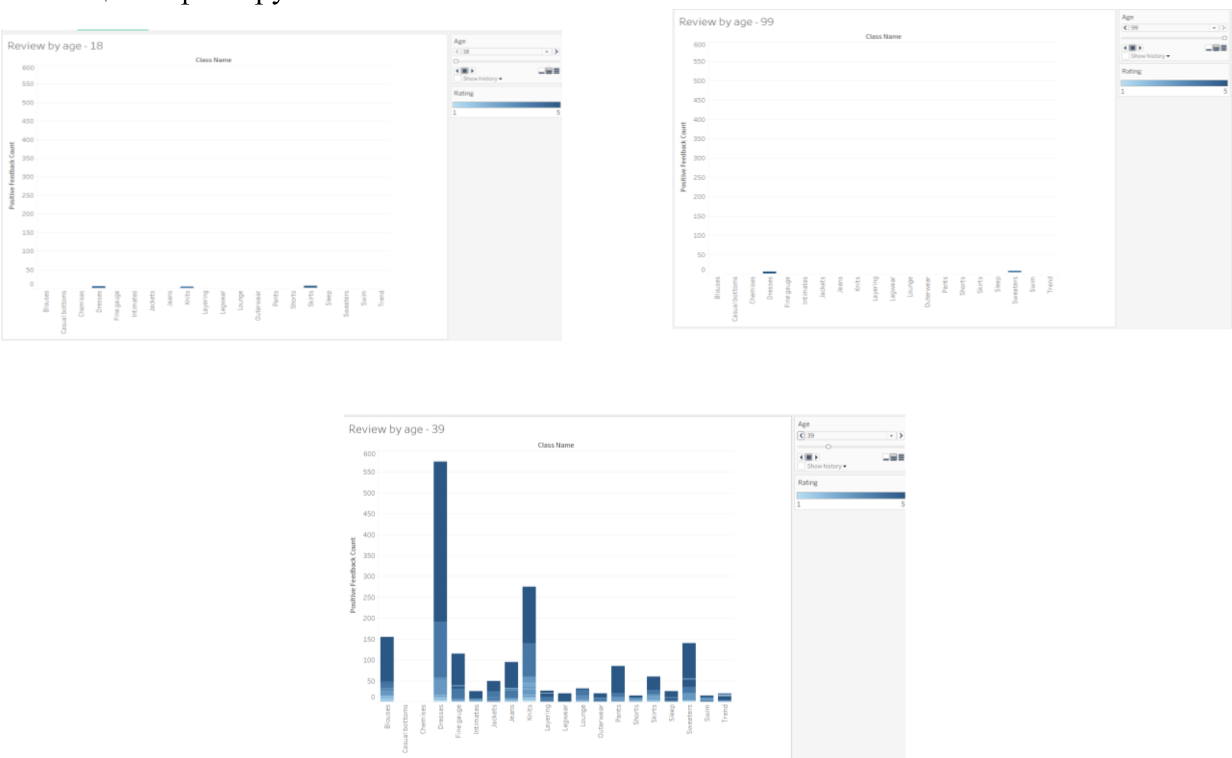


Рис. 2. Приклади аналізу відгуків клієнтів залежно від віку

У цьому випадку (рис. 2) можна спостерігати цікаву тенденцію: найбільша активність спостерігається у середньому віці, хоча на перший погляд здавалося, що молодь мала б бути більш активна у формуванні як позитивних, так і негативних відгуків.

Висновки

У статті розроблено метод аналізу дописів про вподобання клієнтів та розроблено алгоритм на його основі, що дає змогу здійснювати маркетингове дослідження ринку, а також стати основою для здійснення прогнозів продажів на наступні періоди. Особливістю розробленого методу є послідовна комбінація різних методів векторизації текстів, що дає змогу зберегти семантику тексту та відшукати значущі ознаки у ньому. Окрім того, послідовність застосування різних векторизаторів дає змогу будувати ієрархію ознак та маркерів.

Література

1. Pang, B., & Lee, L. (2008). *Opinion mining and sentiment analysis*. Foundations and Trends® in Information Retrieval, 2(1–2), 1-135.
2. Liu, B., & Zhang, L. (2012). *A survey of opinion mining and sentiment analysis*. In Mining text data (pp. 415-463). Springer, Boston, MA
3. Данилюк, І.Г. *Технологія автоматичного визначення тематики тексту* (2008) [Текст]. Лінгвістичні студії: Зб. наук. праць. Випуск 17 / Укл.: Анатолій Загнітко (наук. ред.) та ін. – Донецьк : ДонНУ. – С. 290-293
4. Kolcz, A. *Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization* (2004) / A. Kolcz, A. Chowdhury, J. Alspector. // KDD.
5. Broder, Andrei Z. *Identifying and Filtering Near-Duplicate Documents* (2000). Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (COM'00). – P. 1-10.
6. J. Pennington R. Socher Chr. D. Manning. *GloVe: Global Vectors for Word Representation*. Available from: https://nlp.stanford.edu/pubs/glove.pdf

References

1. Pang, B., & Lee, L. (2008). *Opinion mining and sentiment analysis*. Foundations and Trends® in Information Retrieval, 2(1–2), 1-135.
2. Liu, B., & Zhang, L. (2012). *A survey of opinion mining and sentiment analysis*. In Mining text data (pp. 415-463). Springer, Boston, MA
3. Danyliuk, I.H. *Tekhnolohiia avtomatychnoho vyznachennia tematyky tekstu* (2008) [Tekst]. Linhvistychni studii: Zb. nauk. prats. Vypusk 17 / Ukl.: Anatolii Zahnitko (nauk. red.) ta in. – Donetsk : DonNU. – S. 290-293
4. Kolcz, A. *Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization* (2004) / A. Kolcz, A. Chowdhury, J. Alspector. // KDD.
5. Broder, Andrei Z. *Identifying and Filtering Near-Duplicate Documents* (2000). Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (COM'00). – P. 1-10.
6. J. Pennington R. Socher Chr. D. Manning. *GloVe: Global Vectors for Word Representation*. Available from: https://nlp.stanford.edu/pubs/glove.pdf

RESUME

N. Shakhovska, Kh. Shakhovska

The customer sentiment analysis method from natural language texts

The article presents the method for customer preferences analyzing and develops an algorithm based on it, which enables to conduct market research of the market, as well as become the basis for carrying out sales forecasts for subsequent periods. The peculiarity of the developed method is the consistent combination of different methods of text vectorization, which allows you to keep the semantics of the text and find meaningful signs in it. In addition, the sequencing of the use of different vectors allows you to build a hierarchy of signs and markers.

Based on open data set with Python and Tablau, a software product was developped to analyze customer preferences and visualize the results of analyzes. The use of both tools and the developed method allows for in-depth analysis of natural language texts and reveals hidden data dependencies, such as the age of the client and the nature of his post.

Builded document vectors are very useful, because the sentiment of a sentence can be deduced very precisely from these semantic features. As a matter of fact, users writing reviews with positive or negative sentiments will have completely different ways of composing the words. Feeding a Support vector machine and island clustering with these vectors and training the model to predict sentiment is known to be one

of the best methods for sentiment analysis, both for fine-grained (Very negative / Negative / Neutral / Positive / Very positive) and for more general Negative / Positive classification.

The main method of island clustering is a method that is based on the use of the graph of the coherent occurrence of terms. This graph is based on the definition of the correlation of each pair of terms from which texts are composed, and it is clustered exactly this graph (or part of it) described in the method approach. Thus, the terms of documents are grouped into clusters on the basis of a common occurrence. Also, due to this, this method is resistant to the problems of synonymy and homonymy – terms with different meanings are very likely to fall into different clusters of terms, as they will be encountered in texts in conjunction with different terms. In a pair to this, for the solution of homonymy it is possible to use dictionaries.

Already for the grouping of texts into clusters, based on cluster terms, special procedures are used, which are also described by this method.

Надійшла до редакції 15.10.2018